Remarks/Arguments

The following is a summary of the interview of August 11, 2006

Attendees:

Michael K. Botts, Examiner

Bhupinder Randhawa, Agent

Richard P. Walker, Co-inventor

Christopher Sonnenberg, Co-inventor

Topics Discussed

The parties discussed the Office Action of April 20, 2006 (the "Office Action"), in particular Examiner Botts' objections to the applicant's claims on the basis of Bray et al. (US Patent 6,529,905 B1) ("Bray") and Devanbu (US Patent 6,681,371) ("Devanbu"). The co-inventors and the agent described several differences between the applicant's claims and the prior art.

Conclusions Reached: None

Summary of Discussion

Walker directed Botts to Bray, which forms the principal basis for Botts' objections. Walker pointed out that Bray is tightly bound to the XML document format. Walker directed Botts to the section titled 'Other Publications' on page 1 of Bray, under which one finds a reference to a document titled 'Extensible Markup Language (XML) 1.0: W3C Recommendation Feb. 10 1998', authored by Bray et al. Walker noted that it appears that Bray was intimately involved with drafting the W3C XML document specification.

Walker next directed Botts to Bray column 1 lines 47-50 and pointed out that the XML documentation contemporaneous with Bray's application, namely version 1.0 of the XML specification, is incorporated by reference into Bray.

Walker directed Botts to Bray column 5 lines 6-9 and pointed out that Bray explicitly uses XML to define the allowable relationships between entities. Walker explained that the relationships between nodes in Bray, such as those found in Bray Figures 3 and 4, are defined and controlled by relationships permitted by XML. Walker noted that, as stated by Bray in column 5 lines 4-32, XML documents adhere to a tree structure in which a root node is the parent of the tree, intermediate nodes have one and only one parent but may have many children, and leaf nodes are found at the end of branches.

Walker described the importance of the parent node in XML and Bray. In XML, the parent node stores a link to each and every one of its children. The number and order of child nodes is stored in and maintained by the parent node. To add a new child node in XML, the parent node must be accessed because the parent node needs to establish a link to the new child node and it needs to know how many children it has and also the order of children. In order to delete a child node in XML, the parent node must similarly be accessed. Walker noted that programming APIs that are commonly used to access XML documents, such as Microsoft .NET and Java, target the parent node whenever child nodes are added, deleted, prepended, appended or traversed.

Walker noted that the critical importance of the parent node in XML documents is clearly reflected in Bray's locking rules for adding new child nodes and deleting existing child nodes. With respect to adding new child nodes, Walker directed Botts to Bray column 8 lines lines 4-5, lines 18-23, and lines 35-36. Walker noted that, in these sections, Bray's 'create lock' restrictions precisely follow the XML specification. The parent node is the target node and must be lockable by a user for that user to add a new child node; the parent node maintains a vector of children that is affected by the insertion of a new child node; and a child node may not be added if the parent node is locked in any way.

With respect to deleting existing child nodes, Walker directed Botts to Bray column 8 lines 64-67, column 9 lines 20-27 and lines 58-61. Walker noted that Bray's 'delete lock' restrictions similarly strictly adhere to the XML specification. The parent node must be lockable by a user for that user to delete an existing child node; the number and order of child nodes is maintained by the parent node's vector of children; and a child node may not be deleted if a lock of any kind exists on the parent or the child.

Walker directed Botts' attention to the applicant's claim 41. Walker noted that the parent container does not store a link to any of its children except the first and last sibling containers in the linked list of sibling containers. The parent container does not maintain the number and order of its children. Child containers may be added or deleted to a document subtree without accessing the parent container in any way. The parent container may be unlocked or locked by any user for editing while others add or delete children.

Walker noted that the document structure in claim 41 is illegal under XML and Bray. The 'previous' and 'next' links between sibling containers in claim 41 are prohibited under and unsupported by XML. In XML and Bray, a child node has no direct knowledge or awareness of its siblings.

Walker proceeded to explain the importance of these differences and noted that claim 41 forms the basis of simultaneous multi-user co-editing.

Walker explained that the locking restrictions imposed by Bray essentially preclude any notion of simultaneous co-editing. In Bray, if user A wishes to add or delete a child node, he must lock down the parent. If the parent is locked by another user, he sits and waits until he can gain the parent lock. If user B wants to add or delete a child node, he or she must wait until user A explicitly relinquishes the lock on the parent before he's able to add or delete children. Walker noted that the requirement that the parent node be available for locking when adding or deleting children is a real bottleneck that slows co-editing to a crawl. Walker noted that Bray does not facilitate simultaneous co-editing.

Walker noted that claim 41 moves away from Bray and towards simultaneous co-editing. Walker stated that, during a co-editing session as contemplated by claim 41, a document may have hundreds or thousands of child sibling containers and dozens of active users (co-editors). As a document is co-edited, child containers are added and deleted at a furious pace. To achieve any semblance of simultaneity, or full concurrency, users must be able to add and delete child containers with an absolute minimum of restrictions. In

claim 41, the barest minimum number of containers is locked in order to prevent co-editing conflicts.

Walker noted that in the applicant's claims, the first user can add or delete child containers at will. The parent container is neither accessed nor locked down. The second user can add or delete child containers at will. The parent container is neither accessed nor locked. The parent container may be unlocked, or, as in claim 60, it may be locked for editing by another user. Addition or deletion of child containers takes place independent of the lock state of the parent container. Walker noted that if one were to use Bray's approach to co-edit a document, one would spend a great deal of time waiting for other users to relinquish locks on parent nodes.

Walker next directed Botts' to Devanbu, upon which Botts raised additional objections.

Walker explained that Devanbu deals with 'part editing' in a multi-user domain. Walker directed Botts to Devanbu column 5, lines 1-18, wherein Devanbu enumerates various levels of concurrency ranging from no concurrency to full concurrency. Walker noted that Devanbu argues against full concurrency at column 5 lines 18-25 and again at lines 33-34. Devanbu advocates the lowest level of concurrency and states that full concurrency should be rarely used as he suspects that it demands a great deal of network traffic.

Walker noted that the applicant's claim 41 is the basis of a fully concurrent editor and by its very nature enables the highest level of concurrency.

Walker noted that Devanbu, at the bottom of column 5, describes the work that would be necessary to create a client part editor without listing or describing any of the required "about 60" virtual functions. Walker noted that Devanbu column 6 lines 1-2 states that such a contemplated editor would not deal with concurrency.

Walker directed Botts to Devanbu column 6 lines 6-36 wherein Devanbu briefly discusses creating wrappers for part editors and describes intercepting screen repaint calls in a single-user concurrency model. Walker noted that Devanbu acknowledges, at column 6 lines 37-43, that wrappers may not work for higher levels of concurrency and that the

editor itself may need to be rewritten. Walker noted that Devanbu fails to teach how an editor is wrappered or rewritten to support full concurrency.

Botts asked if a fully concurrent editor, achieved by either wrappering or rewriting an existing editor, is something that someone with reasonable skill at the time of the application would be able to routinely write. Walker and Sonnenberg responded in the negative. Walker noted that wrappering an existing application such as Microsoft Word to enable full concurrency is a staggeringly complex task that would involve intercepting and dealing with an immensely complex variety of application and operating system events. Walker noted that a novel approach, such as that found in the applicant's claims, is required. Walker noted that to this day, Microsoft still does not offer a product that allows two or more users to simultaneously edit a Word document. Botts responded that there are many things that Microsoft does not offer.

Walker noted that that Devanbu completely failed to address co-editing conflicts in a fully concurrent model. Walker noted that, in a multi-user environment or domain, if user A and user B are both editing the same document at the same time, a mechanism must be in place to resolve situations in which both users are editing the same section at the same time. Walker noted that, in contemporary systems such as Lotus Notes, this usually happens when asynchronously editing documents are merged - the merging user usually resolves overlapping co-edited sections. Walker noted that Devanbu simply does not address co-editing conflicts nor does he suggest any method of resolving them in a fully concurrent model.

Walker noted that Devanbu describes sending a 'part modification signal' to other clients at column 4, lines 4-6. Walker noted that Devanbu does not suggest a structure for such a signal, nor does he describe its contents, how or when it is sent, how the receiving client responds to it if at all, or what level of concurrency, if any, is achieved. Walker noted that Devanbu clearly believes that, whatever the nature of such a signal, it must "demand a great deal of network traffic", as stated in column 5 line 24.

Walker directed Botts to Devanbu column 6 lines 60-65 and noted that Devanbu vaguely states that changes are shared with other users "at about the time the change is made".

Walker noted that Devanbu clearly leads in a direction away from our claim 41. Walker noted that Devanbu fails to suggest or teach how to implement higher levels of concurrency and counsels against it. Walker noted that the update messages described in the applicant's claims are extremely lightweight, use very little network traffic, and are essential for fully concurrent simultaneous co-editing.

Walker noted that Devanbu does not suggest or teach any component that might be combined with Bray to achieve the fully concurrent, conflict-savvy co-editing presented in claim 41. Walker noted that if one were to combine Bray, with its restrictive locking rules, with Devanbu, which neither advocates nor describes any details of full concurrency, one would end up with a highly asynchronous system of document editing.

Botts asked whether our claims were intended to address systems that allow a user to sign a document out of some form of database, edit it, then sign it back in. Botts stated that this form of editing is frequently seen in systems such as revision control systems and concurrent versions systems. Sonnenberg responded that these systems generally operate on whole documents and do not support fully concurrent multi-user co-editing. A user 'checks out' a document, edits it, then 'checks it back in'. Other users are subsequently informed that changes have occurred and may refresh their view of the document or project. Walker noted that our claims are not intended to address the lock-edit-unlock scenario. Randhawa stated that we are willing to craft claim 41 to reflect this more clearly. Botts stated that this form of document editing has been addressed by several patents and encouraged our response to the Office Action to clearly differentiate our claims from such systems.

Respectfully submitted,
Richard Walker et al.

Bhupinder Randhawa

Bhupinder Randhawa, No. 47276
Bereskin & Parr, Customer No. 001059